**The Vikings**

**Team Project Sprint 2 (TP-3.2)**

**Hotel Checking System (HCS)**

**Team members:**

**Biniam G Haile**

**Darius O. Nyaundi**

**Elvis A. Ndenge**

**Patrice A. Yemmene**

**Eyad Shesl**

## Introduction

This is the second Sprint of the third Team Project. In this s print, we begin the first iteration of the implementation of the Hotel Checking System (HCS), following the Unified Process described by Graig Larman in his book: Applying UML and Patterns. Larman defines the Unified Process as an “iterative software development Process for building object oriented systems.

In this sprint version of the project, we write use cases, identify External Systems Events, then implement several of them, guided by GRASP (General responsibility assignment software patterns)are refining the glossary (from previous sprint), as well as the Domain Class List. In addition to this, we draw a corresponding Domain Class Diagram (DoCD), before doing the system development.

## Use cases

UC 1: Check room status

**Scope**: Hotel Checking System  
**Level**: User goal  
**Primary Actor**: Front Desk worker  
**Stakeholder and Interest**:

* Front desk worker: wants accurate, fast entry and no checking errors of hotel guests
* House Keeping worker: wants to ensure cleanliness or rooms at the time of booking
* Guest: wants his checking process to be as smooth as possible, and his membership status taken into account
* Hotel: Wants smooth and efficient operations

**Preconditions**: Patron shows up, and confirmation number is validated

Post condition: Guest is checking in successfully

Main success scenario:

1. Guest walks to the front desk, and request checking
2. Front desk worker request booking confirmation number
3. Start a new session, and enters booking confirmation number into the system for validation
4. System validates the confirmation number
5. System will verify available, and clean room
6. System presents available rooms, based on Guest membership
7. Front desk worker discusses room availability with Patrons
8. Patron makes a choice
9. Front desk worker selects room choice
10. System closes the booking process and prints a room tag for customer
11. Customer heads to their room

**Extensions**:

\*a. At any time, Manager request an override of an operation

- System enters Manager-Authorized mode  
- manager of front desk employee performs Manager-mode operation (e.g, void a booking, assign room for silver gold membership to customer who only has silver membership for customer satisfaction reasons …)

- System revers to Front End worker mode

\*b. At any time, the system fails

To support and correct any booking, ensure al transactions sensitive state and events can be recovered from any step of the scenario

* Front desk worker restarts the system, and request recovery of prior state
* System reconstructs prior state
  + System detects any anomalies preventing recovery  
     - System signals error to Front End worker, record the error, and enters a clean state
  + Cashier starts a new checking transaction

Use case 1: Verify room availability and cleanliness

**Scope**: Hotel Checking System  
**Level**: User goal  
**Primary Actor**: House Keeping  
**Stakeholder and Interest**:

* Front desk worker: wants accurate, fast entry and no checking errors of hotel guests, ensuring they are checked into clean and available room
* House Keeping worker: wants to ensure cleanliness or rooms at the time of booking
* Guest: wants his checking process to be as smooth as possible, and his membership status taken into account
* Hotel: Wants smooth and efficient operations

**Preconditions**: Patron shows up, and confirmation number is validated

Post condition: Guest is checking in successfully

Main success scenario:

1. Guest walks to the front desk, and request checking
2. Front desk worker request booking confirmation number
3. Start a new session, and enters booking confirmation number into the system for validation
4. System validates the confirmation number
5. Housekeeping System verify available, and cleanliness of room
6. Housekeeping presents available rooms, based on Guest membership
7. Front desk worker discusses room availability with Patrons
8. Patron makes a choice
9. Front desk worker selects room choice
10. System closes the booking process and prints a room tag for customer
11. Customer heads to their room

**Extensions**:

\*a. At any time, Manager request an override of an operation

- System enters Manager-Authorized mode  
- manager of front desk employee performs Manager-mode operation (e.g, void a booking, assign room for silver gold membership to customer who only has silver membership for customer satisfaction reasons …)

- System revers to Front End worker mode

\*b. At any time, the system fails

To support and correct any booking, ensure al transactions sensitive state and events can be recovered from any step of the scenario

* Front desk worker restarts the system, and request recovery of prior state
* System reconstructs prior state
  + System detects any anomalies preventing recovery  
     - System signals error to Front End worker, record the error, and enters a clean state
  + Cashier starts a new checking transaction

**Scope**: Hotel Checking System  
**Level**: User goal  
**Primary Actor**: Front Desk worker  
**Stakeholder and Interest**:

* Front desk worker: wants accurate, fast entry and no checking errors of hotel guests
* House Keeping worker: wants to ensure cleanliness or rooms at the time of booking
* Guest: wants his checking process to be as smooth as possible, and his membership status taken into account
* Hotel: Wants smooth and efficient operations

**Preconditions**: Patron shows up, and confirmation number is validated

Post condition: Guest is checking in successfully

Main success scenario:

1. Guest walks to the front desk, and request checking
2. Front desk worker request booking confirmation number
3. Start a new session, and enters booking confirmation number into the system for validation
4. System validates the confirmation number
5. System will verify available, and clean room
6. System presents available rooms, based on Guest membership
7. Front desk worker discusses room availability with Patrons
8. Patron makes a choice
9. Front desk worker selects room choice
10. System closes the booking process and prints a room tag for customer
11. Customer heads to their room

**Extensions**:

\*a. At any time, Manager request an override of an operation

- System enters Manager-Authorized mode  
- manager of front desk employee performs Manager-mode operation (e.g, void a booking, assign room for silver gold membership to customer who only has silver membership for customer satisfaction reasons …)

- System revers to Front End worker mode

\*b. At any time, the system fails

To support and correct any booking, ensure al transactions sensitive state and events can be recovered from any step of the scenario

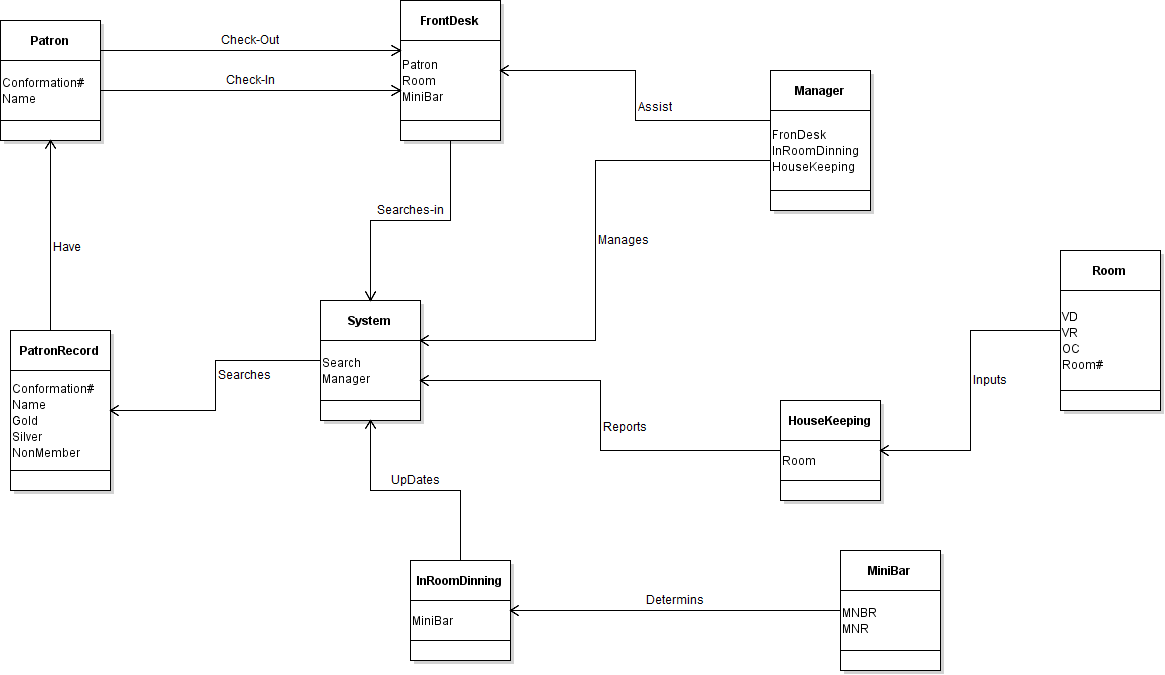
* Front desk worker restarts the system, and request recovery of prior state
* System reconstructs prior state
  + System detects any anomalies preventing recovery  
     - System signals error to Front End worker, record the error, and enters a clean state
  + Cashier starts a new checking transaction

## Glossary

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Term | Description | Format | Range of Values | Validation Rules | Relationship to other elements | Aliases |
| FD | A hotel front desk agent represents the first point of contact with guests and handles all stages of a guest's stay. |  |  |  | Patron, FrontDesk, In-Room dinning, HouseKeeping, System | Front Desk |
| Patron | a customer of the hotel and a occupier of the Rooms |  |  |  | FrontDesk, Silver, Gold, None Member, Rooms |  |
| HK | are responsible for making sure all assigned areas of the room are clean, neat, and tidy. |  |  |  | Room, FrontDesk | HouseKeeping |
| MB | a refrigerator in a hotel room containing a selection of refreshments that are charged for on the bill if used by the occupant |  |  |  | In-Room Dinning | MiniBar |
| Manager | responsible for planning, directing and overseeing the operations |  |  |  | System |  |
| Gold | First Priority highest level of membership. |  |  |  | Patron |  |
| Silver | Second Priority second highest level of membership. |  |  |  | Patron |  |
| NM | Lowest of Priority not a in membership status. |  |  |  | Patron | None-Member |
| Transaction | an exchange or [transfer](https://www.merriam-webster.com/dictionary/transfer) of services |  |  |  | FrontDesk |  |
| Room | Space that can be occupied by Patrons. |  |  |  | HouseKeeping, Patron |  |
| VC | Status of the Room is empty and clean. |  |  |  | Room, Housekeeping, FrontDesk | Vacant-Clean |
| VD | Status of the Room is empty and Dirty. |  |  |  | Room, Housekeeping, FrontDesk | Vacant-Dirty |
| MBR | The MiniBar items are all restocked in MiniBar. |  |  |  | In-Room Dinning, FrontDesk | MiniBar-Restocked |
| MBNR | The MiniBar items are not all restocked in MiniBar. |  |  |  | In-Room Dinning, FrontDesk | MiniBar- Not Restocked |
| Interface | Allow for interaction between another system, person, organization, etc.. |  |  |  | All-CLasses |  |
| IRD | Takes care of all hotel food items in the Room. |  |  |  | MiniBar, FrontDesk | In-Room Dinning |
| System | a set of connected things or parts forming a complex whole, in particular. |  |  |  | FrontDesk, HouseKeeping, In-Room Dinning |  |
| RN | Gives the number of the room. |  |  |  | FrontDesk and HouseKeeping. | RoomNumber |
| ER | If a guest booked prior to stay. |  |  |  | FrontDesk, Manager, system. | ExistingReservations |
| CN | The number a guest receives once booked the room to identify that they are the ones who booked the room. |  |  |  | System, FrontDesk, Patron, System, Manager. | ConformationNunber |
| AN | Number of rooms available. |  |  |  | Manager, Rooms, FronDesk, System. | AvailabilityNumber |
| OC | Once room has a booking already on it or once someone is checked into room. |  |  |  | Rooms | Occupied |
| PR | Contains name and conformation of patron |  |  |  | FrontDesk, System. | PatronRecord |

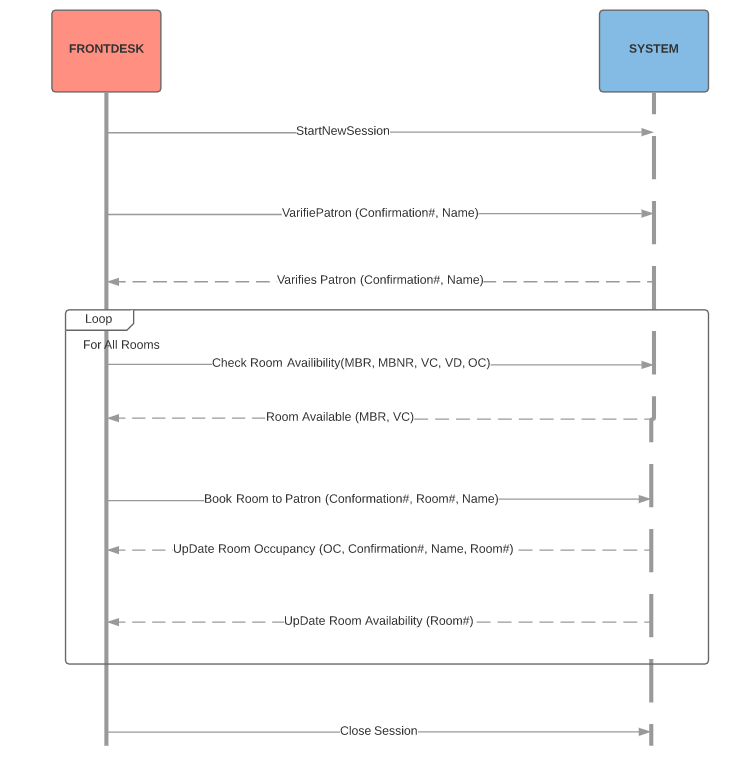
## CLASS DOMAIN LIST FINAL

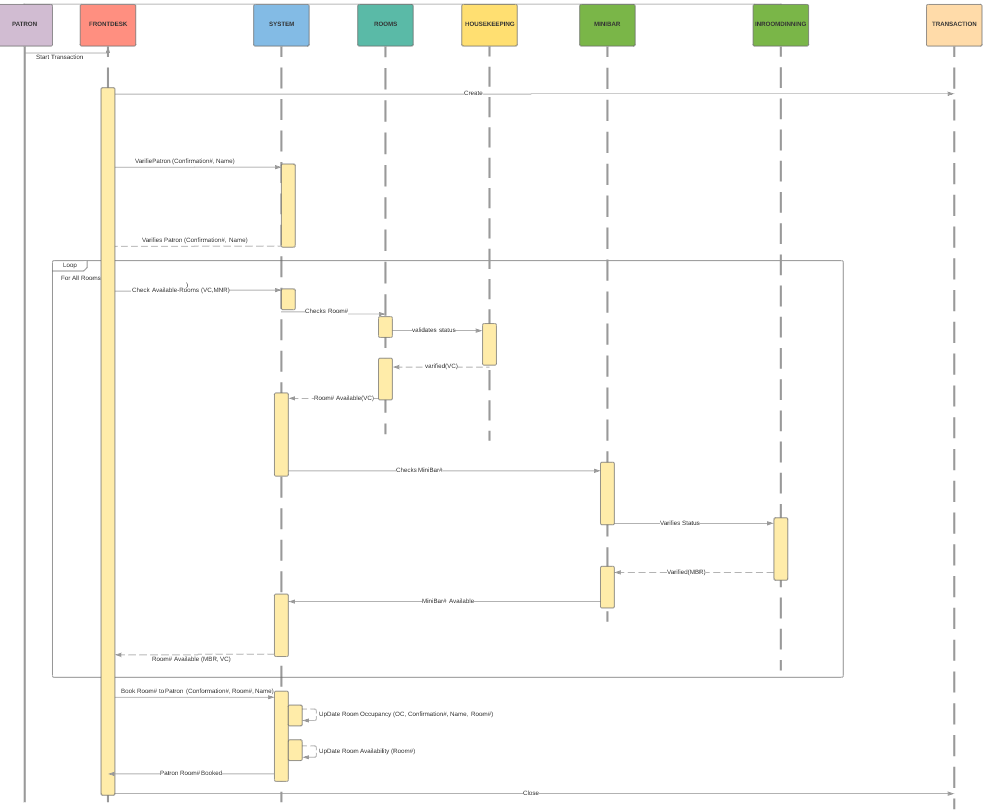
|  |  |
| --- | --- |
| Patron | Customer of the hotel and the occupier of the <<Room>>. |
| HouseKeeping | Checking of <<Room>> status and reporting status to <<FrontDesk>>. |
| InRoomDinning | Checking of <<Minibar>> and reports status to <<FrontDesk>>. |
| FrontDesk | **<<**ConformationNumber>> holder and <<Room>> status holder. Holds PatronRecord as well. |
| Rooms | identifies what status room is in VC, VD, AN. |
| MiniBar | Holds records ofMBNR, MBR |
| HCSApp | Containsinterface/ management/ system. |
| PatronRecord | Holds Name, membership status and ConformationNumber of patron. |
| Confirmation | Holds Patron details, as well as confirmation number |



Patron  
- FirstName  
- LastName  
- Adress  
- Tel:  
- BookedDate  
-PatronId

Confirmation class  
- Status  
- patronFisrtName  
- patronLastName  
- confirmationDate  
- confirmationId





## System Implementation

We develop this system using eclipse IDE and java Eclipse. Eclipse is an open-source software Integrated Development Environment (IDE) for Java developers, consisting of the Java Development Tools (JDT) and the Eclipse Compiler for Java (ECJ)

Java is a very powerful open source (as well) Object oriented programming language. We use Git/EGit and GithUb for code development, committing to a git repository on a regular. We also write jUnit Test cases striving for complete coverage and our Non-UI code

Separation of UI (Presentation Logic) fromDomain Business Logic

## UML class diagram generated by reverse engineering

This is done by using ObjectAid UML